PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	:	
Arnon Amir et al.	:	Group Art Unit: 2173
Serial No.: 09/822,035	:	Examiner: Brian J. Detwiler
Filed: March 29, 2001	:	San Jose, California

Title: **VIDEO AND MULTIMEDIA BROWSING WHILE SWITCHING BETWEEN VIEWS**

DECLARATION UNDER 37 C.F.R. §1.131

Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Dear Sir:

We, Arnon Amir, Dulce Ponceleon, and Savitha Srinivasan, do hereby state that:

1) We are the inventors of the subject matter in the above-referenced patent application.

2) The paper included as Exhibit 1 appeared in the conference proceedings of the Third WWW9 Workshop on Web Engineering, which was held in Amsterdam, The Netherlands, on May 15, 2000. These conference proceedings, including our paper, were published the following year. Our paper (Exhibit 1) appears in print as S. Srinivasan et al., "Engineering the Web for Multimedia" in LNCS vol. 2016, pp. 77-89, S. Murugesan and Y. Deshpande (editors), Springer-Verlag, 2001.

3) The "Guidelines for Authors—Final Submission" (included as Exhibit 2) distributed in connection with this conference required that both a final manuscript and an abstract be submitted to San Murugesan (one of the conference co-chairs) by April 26, 2000, which we did.

4) In an e-mail to us dated April 1, 2000 (see Exhibit 3), Yogesh Deshpande (the other conference co-chair) acknowledged receipt of our paper and invited us to make certain changes to it.

5) An e-mail dated May 9, 2000 from San Murugesan to the conference presenters is included as Exhibit 4. This e-mail includes two attachments, the conference program ("WWW9-WebE-program.pdf") and a listing of the abstracts ("WWW9-Abstracts-Proc.pdf"), which are included as Exhibits 5 and 6, respectively.

6) The conference program (Exhibit 5) shows that we presented a talk titled "Engineering the Web for Multimedia" on May 15, 2000, from 11 a.m. to 12:30 p.m.

7) The second page of the listing of the abstracts (Exhibit 6) includes the abstract corresponding to our talk. This abstract is identical to the abstract that was published in connection with the full paper (Exhibit 1). This abstract is also available on-line at: http://web.archive.org/web/20010215043754/fistserv.macarthur.uws.edu.au/webe2000/WWW9-Abstracts-Proc.htm

8) The paper included as Exhibit 1 is directed to various web-based applications, including the subject matter claimed in the above-referenced application.

9) Section 3.1 of the paper (Exhibit 1) beginning with the middle paragraph on page 84 and continuing through page 85 until the beginning of section "3.2 Related issues" describes switching from one media representation to another. This paragraph reads in part "Consider a "smart" browsing interface that incorporates several representations of a given video such as a slide show, compressed audio, original video, etc. [2, 14] An example of advanced GUI functionality in this case may be that of video controls to *switch* viewing from one representation to another, starting from the time offset of the previously playing video...Figure 3 illustrates JavaScript functions present at the client for providing synchronous playback based on asynchronous streaming video APIs."

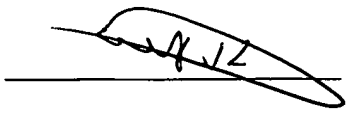
10) Additional discussion in the paper (Exhibit 1) related to switching between views appears in Section 4 titled "Case Study: CueVideo". This section discusses a certain reduction to practice of the invention. The first paragraph of Section 4 includes the following sentence: "As an application, we would categorize this as a fairly advanced web-based video application which allows searching of video collections, playback of relevant video segments, ability to start playback of a new representation of the video based on current playback position, etc..." Also, the second sentence on page 87 refers to "advanced player functionality to support switching of video from one source to another based on user action".

11) Section 3.1 generally, Section 4 generally, and Figure 3 in particular of the paper (Exhibit 1) demonstrate that we had reduced to practice methods directed to the synchronous switching of a first media stream to a second media stream prior to November 11, 2000. Further, Figures 1 and 2 of the paper (Exhibit 1) and the associated discussion illustrate how the invention can be viewed from the either the user's perspective (as in Claim 1, for example) or from the provider's perspective (as in Claim 27, for example).

12) The sentence in Section 3.1 of the paper (Exhibit 1) that reads "Consider a "smart" browsing interface that incorporates several representations of a given video such as a slide show, compressed audio, original video, etc." demonstrates that the first and second media streams may have media content in common with each other (as in Claims 2 and 28, for example). This sentence also demonstrates that the media stream could be full video (as in Claims 3, 16, 29, and 36, for example) or audio (as in Claims 4, 11, 12, 30, and 34, for example).

13) The sentence in Section 3.1 of the paper (Exhibit 1) that reads "An example of advanced GUI functionality in this case may be that of video controls to *switch* viewing from one representation to another, starting from the time offset of the previously playing video" demonstrates that playback of the first media stream may end at a point in time where the playback of the second media stream begins (as in Claims 8, 9, and 32, for example).

We hereby declare that all statements made here of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application of any patent issued thereon.

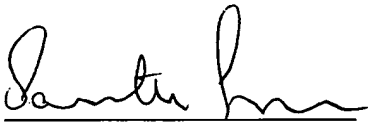


June 2nd, 2004

Arnon Amir — Date

May 28, 2004

Dulce Ponceleon — Date



June 3, 2004

Savitha Srinivasan — Date

ARC920010062US1.131affidavit 1.doc

Engineering the Web for Multimedia

Savitha Srinivasan, Dulce Ponceleon, Arnon Amir, Brian Blanchard, and Dragutin Petkovic

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120 USA
{savitha,dulce,arnon,bblanch,petkovic}@almaden.ibm.com

Abstract. This paper examines the issues related to developing web applications that use digital media, with particular emphasis on digital video. The nature of digital video brings additional complexity to engineering solutions on the web due to the large data sizes in comparison with text, the temporal nature of video, proprietary data formats, and issues related to separation of functionality between content creation, content indexing with associated metadata, and content delivery. The goal of this paper is to contribute to the understanding of different component technologies involved in deploying video-based web applications, and the tradeoffs involved with each option. As an illustrative example, we describe the requirements leading to the architecture of a video-based web application, CueVideo: a system for search and browse of video and related material.

1 Introduction

Web Engineering has been defined as the use of sound scientific, engineering and management principles and disciplined and systematic approaches to the successful development, deployment and maintenance of high quality web-based systems and applications [7]. This paper examines how multimedia, and in particular digital video is supported in a typical web application. We describe the issues related to the usage of digital video in applications, and summarize the current state-of-the-art technologies that enable the deployment of video in web applications. Finally, we describe the components of a specific web application deploying video, and illustrate the reasons that lead to this particular architecture.

Today, large collections of multimedia documents can be found in diverse application domains such as the broadcast industry, education, medical imaging, and geographic information systems. Digital video libraries are becoming pervasive since infrastructure needs such as storage requirements, computing power, and high bandwidth networks are being addressed adequately. Video streaming technology allows content to be delivered to the user as a continuous flow of data with minimal wait time before playback, such that the user does not need to partially or fully download the video files before starting to view the video. Further, a variety of video

S. Murugesan and Y. Deshpande (Eds.): WebEngineering 2000, LNCS 2016, pp. 77-89, 2001.

editing, capture and compression tools are available to facilitate the manipulation of digital video. Seemingly then, all component technologies necessary for deploying web applications containing video are available. However, as we start to develop such an application, several architectural issues arise that are unique to the characteristics of digital video. These include partitioning of video related application functionality between front-end and back-end, and how the video control program logic gets to the front-end. Another source of the problem is the application functionality and selection of competing transport protocols, video compression codecs and containers on the web. For example, there exist several standard transport protocols, and video compression codecs; however, there is no universal consensus in their usage and proprietary alternatives are often used. This leads to incompatibilities between platforms and browsers, and requires additional development to support web applications deploying video.

Typical web applications may be characterized as adhering to a three-tier architecture. The front-end or user interface is delivered to the client by the web server, and is rendered by any HTML capable web browser. The middle-tier consolidates the application specific logic and is invoked by the web server using a standard API. The middle-tier may also connect to the third-tier or back-end database using traditional client-server communication protocols independent of the web. Such an architecture enables the separation of the GUI from the application specific logic. One issue specific to video-based web applications is that typically searchable metadata associated with video is often kept separate from the servers that actually deliver video. For example, the metadata may reside in the third-tier such as a relational database, while the video may be streamed from a streaming media server. The streaming media server does not strictly belong to the third-tier since it directly streams media to the client. It may be seen as a new extension to the middle-tier, since it does not fit into the application server directly.

We view the process of developing web applications that deploy video as having four components to it:

1. Content creation including generation of digital video and authoring multimedia presentations
2. Content cataloging and indexing tools that create searchable metadata associated with the media
3. Design of search and browse system that includes search engine and user interface design
4. Content delivery and presentation

The challenges associated with all four components listed above currently serve as impediments to the widespread usage of video. Certain operational aspects related to this such as streaming media technologies, and optimization algorithms for efficient content delivery on the web are being adequately addressed by advances in technology. In this paper, we focus on software engineering issues related to the fourth component involving multimedia content delivery and presentation on the web. First, we describe general issues related to streaming media on the web, and second, we describe specific application issues related to developing advanced streaming media functionality on the web.

2 Streaming Media

Streaming media is a necessary component of all web applications that deploy video. There are several factors that must be considered when selecting a particular streaming media technology. We describe some of those factors.

2.1 Media Player Functionality

The general requirements for any streaming video player are reliability, support on multiple platforms, and the availability of an SDK that provides programmable control over the GUI. Consistent with the thin-client model, minimal installation and configuration on the client side is desirable. Video streaming typically provides an application with direct access to any arbitrary location in the video and immediate playback from that point with only a small delay for buffering. For advanced applications, additional requirements may be the ability to switch playback between different videos in the same player window, and the ability to create hotspots or linked regions in the video. Some of these requirements are addressed by the W3C SMIL standard [4] for synchronized multimedia over the web.

The W3C SMIL standard allows the integration of a set of independent multimedia objects into a synchronized multimedia presentation. Since a static web page has no internal timeline, there exists no preset order in which images and text is downloaded. The SMIL format enables a presentation comprising of multiple video/audio clips, images, and text to be synchronized on a single presentation timeline. Using SMIL, an author can describe the temporal behavior of the presentation, describe the layout of the presentation on a screen, and can associate hyperlinks with media objects. Client/server support of SMIL format enables an application to render powerful synchronized multimedia presentations. Today, most video streaming servers and players support the SMIL format based on standard and proprietary formats for compressed video [6, 9, 11, 17].

2.2 Media Player Client Software

Almost all solutions available today are based on browser plug-ins/ActiveX components with varying levels of desired functionality. A plug-in is a unit of code, a shared library or a DLL, which is linked into the web browser which is associated with a MIME type and activated when the user retrieves a resource of that type. It is then given some control over the browsers screen real-estate and can render the resource in whatever way is appropriate. Since plug-ins consist of native code, they are not cross-platform and have full access to all resources of the client host and thus, are not very secure. The plug-in code is usually down-loadable over the web and has to be installed or executed. Hence there is an issue with maintenance and propagation of updates. The other option is Java applets that enable the safe execution of downloadable content. However, this has not been as popular since it requires considerable development effort, has possible performance issues, and continues to rely on native code in some cases. Some Java applet solutions are highly optimized for minimal code size and do not rely on native code. However, the limited code size imposes limits on player performance and functionality.

2.3 Client/Server Platforms

In the area of multimedia, selection of an operating system for a media server proves more challenging than traditional web solution deployment. In an effort to reduce web support costs, companies often select a standard platform (hardware, operating system etc.) for web servers, which may not be equipped to handle media streaming. The selection of both server and client platforms for streaming media have different implications in web applications deploying video:

- Web-browsers are intended to support client applications in a platform-independent manner. This is not entirely true with streaming media applications. Currently, different platforms support a subset of the available streaming formats. For example, the plug-in player that supports QuickTime streaming is available on the Macintosh and Windows platforms only. Similarly, the plug-in player that supports Windows Media streaming is available on the Windows platform only.
- Streaming servers must be deployed on an operating system that supports the latest release of streaming technology in order to take advantage of new features. For example, Apple's current open source QuickTime streaming server is available on the Linux platform only. This means that QuickTime streaming can be deployed only if Linux is selected as the streaming server platform.
- Media servers may incur occasional downtime in order to incorporate latest upgrades. Therefore, streaming media is best hosted on a platform dedicated to providing optimized media functionality without combining it with critical functionality such as transaction databases, home page web servers, etc.

2.4 Bandwidth Considerations

It is necessary to target a network connection bandwidth for delivering a successful streaming media presentation. The content must be created for a particular bandwidth. The goal is to create presentations with bit rates that do not exceed the end-to-end bandwidth (from the streaming media source to the client) in order to avoid loss of data. The target bandwidth is defined to be the maximum bandwidth available for a particular network connection. The total bit rate of the streaming presentation must be at or below the target bandwidth. The total bit rate of a presentation consists of two components: the maximum bit rate consumed by all streaming tracks, and a specific percentage, say 25% [11], of target bandwidth for overhead such as connection noise, data loss, and packet overhead.

There are two approaches to ensure smooth delivery of streamed presentations for varying bandwidths: Some video encoders allow encoding of a single clip that targets multiple bandwidths [11] by including several streams at different bandwidths, and associated synchronization information. Such encoding enables on-the-fly switching to a lower bandwidth encoding under high network traffic to ensure smooth media delivery. The other approach is to create multiple versions of the clips for different bandwidths, and use a SMIL file to designate a target bandwidth for each of the groups when assembling the presentation. Yet another option is to provide low bandwidth representations of the original video based on analysis of the video content. These representations include storyboards, moving storyboards (slide shows) with and without audio, fast playbacks, etc. [2, 14].

Bandwidth considerations also impact the scalability of a web application deploying video. Since the total bandwidth available to a media server is fixed, trade-offs must be made in deciding how that bandwidth is allocated. From the media server's perspective, this may be optimized using a combination of mirrored sites and caching technology. For example, a classroom of 100 students simultaneously accessing the same video clip can be better served with a geographically close mirrored web site and caching technology, than with a mirrored web site alone. From the application's perspective, it will always be possible to exceed the capacity of the bandwidth, despite all enhancements to optimize bandwidth on the server side. In general, a compromise must be made between a smaller number of high bandwidth connections, and a large number of low bandwidth connections.

In summary, the following considerations must be evaluated in selecting a streaming media technology [10]:

- Streaming media formats supported by browser plug-in player
- Browser installation: plug-in versus applet
- Availability of SDK with API access, ease of development
- Tools for creation of rich media content
- Support for SMIL format to provide spatio-temporal synchronization of media at presentation time
- Target platform of both streaming media server and client (e.g. Linux, Windows, Macintosh, set-top box, etc.)

3 Streaming Media Applications

Today, web applications have taken on a lot more complexity than merely being static sources of information. They include e-business applications, distance learning and training, enterprise-wide planning systems, transactional systems, collaborative work environments, etc. The same parallel is seen in web-based video applications: they vary from being relatively static information sources (as in streaming short, well edited video clips) to increasingly complex applications such as e-business and transactional systems that must support highly interactive, searchable, large collections of media.

In its simplest form, a single digital video may be delivered using streaming video technology. This means a user can play the video clip from start to end, and have access to VCR-like controls such as Play/Pause/Stop/Forward/Rewind for browsing/viewing the video. This is often the case with video clips in broadcast news web sites. Such applications are adequately handled by most of the available video streaming client/server architectures. The video content is typically edited, and prepared manually to be effective, short and most appealing to the user. In contrast, an advanced application may be one where the streaming video is highly interactive and searchable, and can be of much longer (tens of minutes or more). Such applications are powered by video cataloging and indexing tools [2, 3, 13, 14, 16] that enable sophisticated search algorithms to make video searchable. Another example of an advanced video-based web application is one where user navigation triggers database search, as a result of which streaming media is dynamically composed into a web

page so as to display contextual media to the user. Key issues like the degree to which the web-based client contains intelligence, and how that intelligence was made available to the front end are very much dependent on the application functionality. In general, tradeoffs are made between desired functionality and performance. Simple applications that essentially provide a few short streaming video clips with standard VCR-like controls for playback can be architected with minimal intelligence at the front end. In contrast, highly interactive applications that include media search, and intelligent content-based video representations may involve some program logic at the client for performance reasons.

3.1 Architectural Options

We examine various architectural options for deploying video in web applications with progressively increasing complexity and functionality. The key issue being illustrated here is the extent of intelligence contained in the client, and the implications of this on the client's processing load and server storage/processing load.

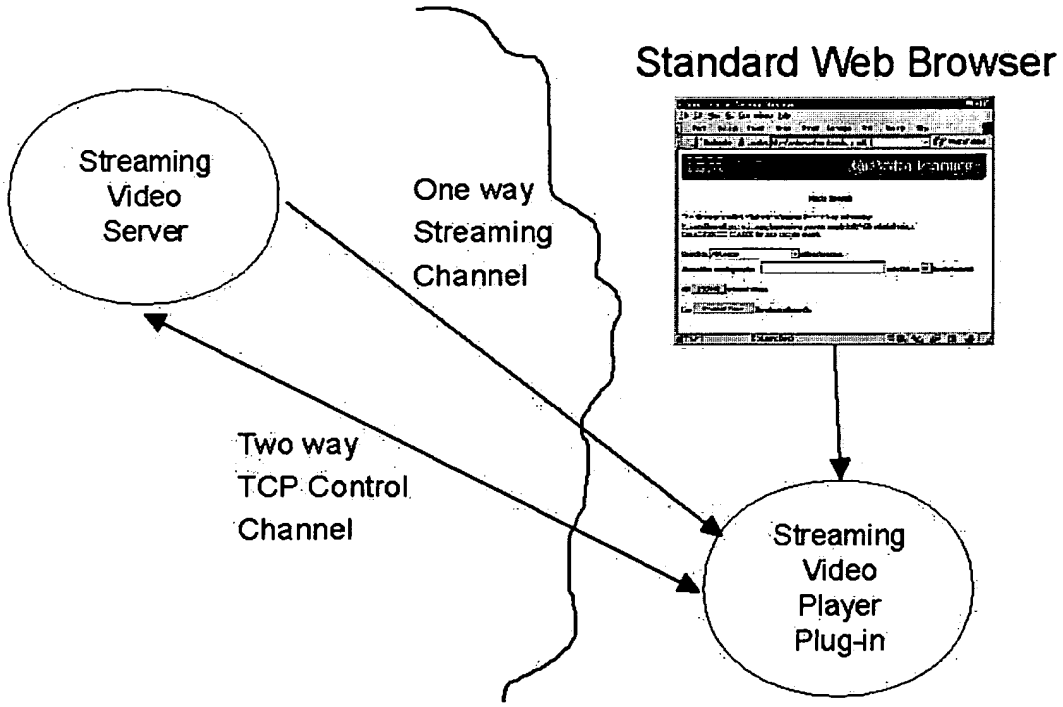


Figure 1: Streaming Video in a Web-based Application

Figure 1 illustrates the architecture of the simplest form of streaming video on the web. The functionality provided is that of playing video on demand with standard VCR-like controls. An HTML page containing a link to a streaming media presentation is rendered in the browser, based on a HTTP request to the web server. A user click on that link causes the player plug-in to establish two connections between the streaming media server and the player plug-in. The first is a two-way TCP connection to send commands from the player to the server such as Play/Pause/Stop, etc. This channel also supports authentication and logging functionality on the server.

The second connection is a one way data channel established from the streaming media server to the player plug-in to stream the media.

Figure 2 illustrates the architecture of a form based two or three-tier web application in the context of video [5]. A simple example is one of typing keywords for search in a video, and retrieval of relevant video segments with the ability to playback the retrieved segments. First, a standard "search" page (in HTML) is presented. It contains a form with fields for typing in keywords for search in a video. The user types in a query and clicks on the submit button. As a result, an HTTP request is sent to the web server, in response to which the application server is invoked. The application server performs the search, and generates the results HTML page containing links to time offsets into the video (based on the query results) to support playback of the relevant video segments. The client can then play a video segment by communicating directly with the video streaming server.

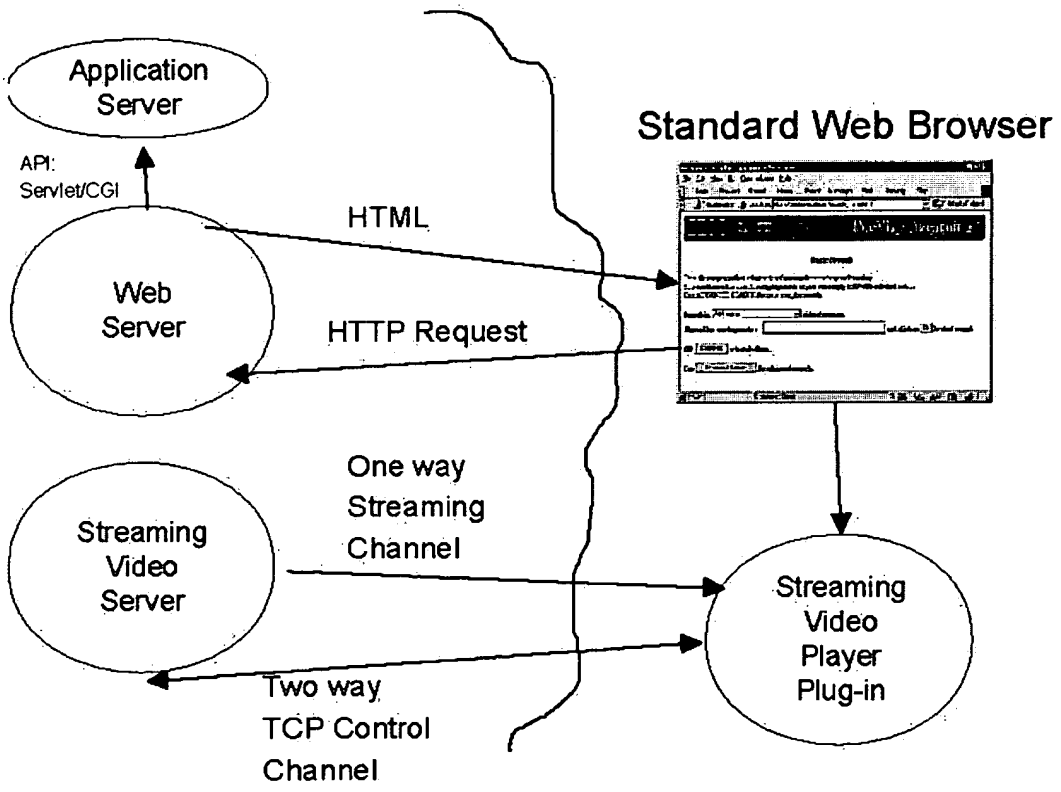


Figure 2: Streaming Video in a Two-tier Web Application

This example illustrates a simple instance of a more general issue, i.e. where should application specific intelligence be generated and reside, at the client or at the server. The location of the query processing to generate the time offsets is not an issue. This adheres to the three-tier architecture in web applications where application specific logic is invoked by the web server using a standard API. The issue is with respect to implementing playback of video with time offsets based on text query results. We have the following options:

- The SMIL standard supports playback of a video with a time offset. On receipt of a query, the application server can generate a new SMIL file with the necessary

time offsets, and refer to this file in the results HTML. A user click on that link will cause the player plug-in to begin playback based on the time offset specified in the SMIL file. This has the disadvantage of having to create several temporary SMIL files for each query, their maintenance, and removal.

- The front-end being rendered by the browser may include program logic to begin playback with the specified time offsets. Most streaming video servers and plug-ins support a limited set of Java/JavaScript APIs for implementing such functionality. Since HTML is primarily a markup language for content presentation, and not for programming logic, the application server must generate the necessary Java/JavaScript code to support the functionality of offset playing based on query results. This adds complexity to the issue of separating program logic from presentation logic at the application server.

While neither option is optimal, the specifics of the application may favor one option over the other.

The described example in figure 2 illustrates the simple GUI functionality of supporting offset playback based on the query results. Within the same framework, certain applications may require advanced GUI functionality with respect to video playback. Consider a "smart" video browsing interface that incorporates several representations of a given video such as a slide show, compressed audio, original video, etc. [2, 14] An example of advanced GUI functionality in this case may be that of video controls to *switch* viewing from one representation to another, starting from the time offset of the previously playing video. Current plug-in players establish stateless sessions where the player maintains no information about prior videos viewed, current position in video, etc. The only option therefore, is to use Java/JavaScript API to implement such functionality. Streaming video servers and plug-ins typically support a set of *asynchronous* APIs without reliable callback support for notification methods. This requires complex programming in JavaScript to provide synchronous functionality based on asynchronous API's to maintain state information in the browser. Providing a robust solution with cross-browser compatibility given the differences in scripting language support and plug-in/ActiveX support proves to be extremely challenging. Figure 3 illustrates JavaScript functions present at the client for providing synchronous playback based on asynchronous streaming video APIs.

Another example of the tradeoffs between the processing power requirements of the client and storage requirements on the server is illustrated in a video browsing application [8]. The study describes three options for streaming content-based video summaries, in this case, a summary based on speedup of audio (and therefore, video) without change in pitch, intonation and overall quality. Such summaries may be precomputed and stored on the server, computed by the server on-the-fly on client request, or computed on the client in real time. Precomputing the summaries on the server requires preprocessing of the audio and storing different versions in several different speeds (depending on the video format, in some cases the entire video must be re-encoded and stored). In this case the client only needs to select the desired representation, i.e. playback speed. However, a synchronization issue (as discussed above) occurs if the client should change the speed during playback. In contrast to this solution, the speedup algorithm may be implemented on the client. This requires additional computation power on the client that is already busy with decoding and

```
function JumpToVideo(newvideo) {
  //Save Position....
  CurntPos = document.Video.GetPosition();
  document.Video.DoStop();
  document.Video.SetSource(newvideo);
  document.Video.DoPlay();

  //WAIT for Play to begin
  myTimer=setTimeout('WaitForPlay()',100);
}

function WaitForPlay () {
 //if NOT yet playing, keep waiting....
  if (document.Video.GetPlayState() != 3)
    myTimer=setTimeout('WaitForPlay()',100);
  else
    {
     document.Video.DoPause();
     document.Video.SetPosition(newpos);
     document.Video.DoPlay();
    }
}
```

Figure 3: JavaScript Functions for Synchronous Playback

presenting the video. It also requires higher streaming rates by the server, since both audio and video must be played faster than the original speed. The overall recommendation in this particular study was to use server storage and precomputed discrete speeds, rather than to overload the client with additional processing and synchronization associated with client control of playback speed.

3.2 Related Issues

So far, we have described specific considerations related to the architecture of web applications deploying streaming video. The web today, is used by millions as a primary source of information and a medium to collaborate, communicate and share knowledge. Information technology in the form of digital libraries has explored issues of organization, access, security, and distributed information sources on the web [12]. This brings out several other issues that must be considered when deploying video in real-world operational systems:

- Separation of second tier (application server with search engine), and third-tier (metadata related to media) from the actual storage and delivery systems that serve the media. The storage and delivery systems may be separate on dedicated servers placed in selected geographical locations or centers with high bandwidth connections. They may also be replicated to ensure better performance. As an example, there exists technology to provide dependable, high-performance delivery of streaming media using sophisticated optimization algorithms for content delivery [1].

- Interoperability: Today, the choice of video formats prevents interoperability among heterogeneous repositories distributed on the web.
- Similarly, the lack of standards for metadata descriptions prevents the use of common search engines across various video databases. The emerging MPEG-7 standard for audiovisual data attempts to address this problem [15].
- Media Search Engines: As video search technology matures, a unified architecture is needed to accommodate different search engines specific to video.
- Media content is typically associated with copyright issues that may influence the way in which search engines work on media. For example, the composition of video summaries from the original video by anyone other than the content owner may be prevented by legal copyright requirements in certain media industries.
- Mirrored web sites: While the use of distributed mirrors increase reliability, accessibility and capacity, it introduces issues of keeping the mirror sites synchronized, and user state synchronization. This is further complicated by the separation of the second and third-tier from the streaming media servers which stores and delivers the video. As with HTTP, the best strategy to address scalability is to deploy caching media servers at client sites.

4 Case Study: CueVideo

In the CueVideo [2, 14] project, we combine computer vision techniques for video content analysis, and speech recognition for spoken document retrieval, together with a user interface designed for rapid filtering and comprehension of video content. We are working on different applications that deploy this technology using a web infrastructure such as distributed learning, just-in-time training, e-business and knowledge mining. As an application, we would categorize this as a fairly advanced web-based video application which allows searching of video collections, playback of relevant video segments, ability to start playback of a new representation of the video-based on current playback position, etc. Our current requirements target an intranet environment with a few hundreds of videos in the collection. Examples of typical content include video recordings of seminars, technical presentations, training tapes, on-line classes, etc. The anticipated number of simultaneous users accessing streaming video is around 25.

4.1 Requirements

Our general set of requirements related to web applications deploying video were as follows:

- Use of standard web technology, with particular emphasis on using standard technology on the client-side. Our solution was not to mandate specialized client setup or technology other than the media player in order to appeal to a large user community.
- Selection of a widely-used video format for streaming compressed video on the web. Since the quality of web-based video is rapidly improving, the selection of a format that is able to capitalize on these improvements was important.

- Video players for the client side that support immediate playback without waiting for entire media file to load, including offset playback to support retrieval of application driven segments. We also needed advanced player functionality to support switching of video from one source to another based on user action.
- Video encoding tools that support standard video formats as inputs and generate standard compressed streaming formats. Accepting input from other standard formats allows an encoder to take maximum advantage of existing repositories of other media types (e.g., MPEG-1, AVI) Additionally, it was necessary for the encoding tools to offer either API access or command line access for automated batch processing.
- As an alternative to converting existing media to a streaming format using video encoding tools, we wanted the option of developing extensions to the base streaming server/player to support new media formats.

4.2 Architectural Solution

We made the following choices to meet our requirements:

- HTTP based HTML applications with a browser plug-in to support video playback functionality in a standard web two-tier application. In addition, we also use Dynamic HTML (DHTML) to support some of the advanced video playback functionality. The three main technologies that make up DHTML are HTML, JavaScript and Cascading Style Sheets (CSS). HTML is used for the basic structure of the document, JavaScript to manipulate the Document Object Model (DOM), and Cascading Style Sheets (CSS) to define the presentation and style of the document. As a result, the new tags supported in HTML trigger additional JavaScript events enabling more control over the content being rendered.
- Thin client model with CGI application layer that allows users to query the videos using different search modes. The advanced video browsing content was precomputed and streamed from the server. However, video control logic such as time offsets, names of appropriate video source files, etc. was generated by the CGI program and encapsulated in DHTML being rendered at the client.
- Streaming video solution using RealNetworks [11]. RealServer has an extensible, component based architecture that allows add-on support for additional media formats such as MPEG-1. The client video playback support is via RealPlayer plug-in. The RealProducer encoding tools enable the creation RealMedia content from a variety of input sources, including live feed, AVI and MPEG-1. It also supports command line processing and API access, allowing smooth integration into applications.

4.3 TradeOffs/Lessons

The tradeoffs associated with our choice of tools and methods were as follows:

- The advantages of HTML are the centralized maintenance, cross-platform nature, simple and consistent user interface, and scalability offered due to caching of HTTP clients. The disadvantage of HTTP based services is that HTTP is stateless and does not support server side initiated communications. Further, HTML

supports only simple user interfaces while advanced GUIs need a richer widget set. Our choice of DHTML to provide additional video player functionality has its disadvantages. DHTML is not yet stable enough to use in mission critical web pages, and cross-browser compatibility due to differences in the document object model between browsers are a big cause for concern. We had to craft our web pages for the different platforms we targeted, in particular Netscape Navigator and Microsoft Internet Explorer for Windows and Macintosh.

- Streaming video is primarily supported with the use of native code using plug-ins for browsers. We chose this option over the Java applet option which supported proprietary formats, consisted of limited functionality, and higher development costs. As a result, clients have to incur the disadvantage of installation of plug-in video players.
- The tradeoffs associated with including video control logic in a dynamically generated HTML page have to do with function and performance. The video control logic in response to user input is generated as JavaScript functions. This contributes to about 6Kb of code that must be served using HTTP for each dynamically generated page. Given the available intranet bandwidth, this overhead was reasonable for the additional functionality gained.
- The tradeoff between quality and bandwidth is a result of the encoding technology, and the method used for bandwidth selection during streaming. For example, the RealNetworks streaming server incorporates the concept of stalling under low bandwidth conditions. That is, when the client does not have sufficient data for playback, the server stops streaming until such time that the client buffers enough data to continue playing smoothly. This feature coupled with seamless switching to a lower bandwidth encoding under low bandwidth (SureStream technology) results in relatively smooth streaming media presentations under unpredictable network traffic conditions. We compromised on the quality of the video with the lower bandwidth encoding, in exchange for the smooth playback functionality without loss of data.

5 Conclusions

In this paper, we have highlighted the architectural considerations for developing web applications deploying video. While there is no clear right answer, we have pointed out the pragmatic issues that must be considered and the options associated with each architectural decision given the available technology. Advanced web-based video applications typically require some amount of computation and player control beyond streaming. In general, the thin client model is favored where the intelligence, and therefore, the processing requirements are at the streaming server. Intelligent interfaces that accompany such applications are rendered at the client, but typically composed at the server by the application. This is a trend, however, the specific goals of any project will dictate the final choice of tools and architecture in deploying a solution.

6 References

1. See URL at http://www.akamai.com/
2. Amir, A., Ponceleon, D., Blanchard, B., Petkovic, D., Srinivasan, S. and G. Cohen, Using Audio Time Scale Modification for Video Browsing. *Proceedings of HICSS-33*, Hawaii, Jan. 2000.
3. Gupta, A. and Jain, R. Visual information retrieval. *Communications of the ACM 40*, 5 (May. 1997), Pages 70 - 79. Also see URL at http://www.virage.com/.
4. Hoschka, P. (ed.). *Synchronized Multimedia Integration Language*, World Wide Web Consortium Recommendation. June 1998. URL: http://www.w3/org/TR/REC-smil.
5. Kristensen, A., Developing HTML Based Web Applications, *Proceedings of the First International Workshop on Web Engineering* at WWW7 Conference, Brisbane, 14 April 1998
6. See URL at http://msdn.microsoft.com/downloads/webtechnology/ie/iepreview.asp
7. Murugesan, S., Y. Deshpande, S. Hansen and A. Ginige, Web Engineering: A New Discipline for Web-Based System Development. *Proceedings of the First ICSE Workshop on Web Engineering at ICSE'99*. 6 -17 May 1999, Los Angeles, USA .
8. Omoigui, N., He, L., Gupta, A., Grudin, J. and Sanocki, E. Time-Compression: Systems Concerns, Usage, and Benefits, *Proceedings of ACM-CHI*, 1999, pp. 136-143.
9. QuickTime Components, Inside Macintosh, by Apple Computer. Also see URL at http://www.apple.com/quicktime/
10. Ramnarayan, S. "Streaming Multimedia: Who is leading?", GartnerGroup, (gartnerweb.com) Jan 18, 1999.
11. RealNetworks, Inc. RealPlayer G2, See URL at: http://www.real.com/products/player/.
12. Schatz, B. and Chen, H. Digital Libraries: Technological Advances and Social Impacts. *Computer* Magazine, Feburuary 1999.
13. See URL at http://speechbot.research.compaq.com/.
14. Srinivasan, S., Ponceleon, D., Amir, A. and Petkovic, D., "What is in that video anyway? In Search of Better Browsing", *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pp. 388-392, Florence, Italy, June 99.
15. van Beek, P., Sezan, I., Ponceleon, D. and Amir, A. Content Description for Efficient Video Navigation, Browsing and Personalization, to appear in *IEEE CVPR Workshop on Content-Based Access of Image and Video Library*, South Carolina, June 13-15, 2000.
16. Wactlar, H., Christel, M., Gong, Y. and Hauptmann, A. Lessons Learned from Building a Terabyte Digital Video Library. *Computer*, February 1999.
17. Yu, J. *A simple, intuitive hypermedia synchronization model and its realization in the browser/Java environment.* Technical Note 1997-027a, Digital Equipment Corporation Systems Research Center, Palo Alto, CA, April 1998. Also see URL at http://www.research.digital.com/SRC/HPA

Exhibit 2

Guidelines for Authors -- Final Submission

Revised submission date: 26 April 2000

As informed earlier, you paper will be included in the forthcoming Springer Verlag's publication on ***Web Engineering***, in the Lecture Notes in Computer Science (Hot Topics) series, both in printed and online versions.

To have consistency and uniformity of the articles and hence enhance quality of the publication **please follow the guidelines strictly.**

Final Manuscript

1. Refer to Information for Authors available at http://www.springer.de/comp/lncs/authors.html

2. Authors' instruction in PDF format is available at http://www.springer.de/comp/lncs/instruct/typeinst.pdf

   This specifies printable area - 122mm x 193 mm SINGLE COLUMN-, layout, typefaces and font sizes.

3. For MS Word users template is available online - see Section "Proceedings and Other Multi-author Volumes" in the information for Authors (Sl No 1 above)

4. Prepare your manuscript according to these guidelines and send two soft copies of your final manuscript one in PDF and the other in RTF.

5. **File name for full manuscript:** The file name has to be the first six letters of Surname of the first author followed by appropriate extension. (Example: alexan.pdf or alexan.rtf when the Surname of the first author is Alexander)

Abstract

6. In addition, separately create a text or Word doc file, consisting of the Title, Authors' Names and organisation, email addresses of authors and the abstract. This will be posted on the Workshop Web page.

7. **File name for Abstract:** The file name has to be the first six letters of Surname of the first author followed "-abst" and appropriate extension. (Example: alexan-abst.tex or alexan-abst.doc when the Surname of the first author is Alexander).

Copyright Form

8. Complete and sign the copyright form available at http://www.springer.de/comp/lncs/copyrigh.html and send it **by Airmail** to: Dr San Murugesan, Dept of Computing and Information Systems, University of Western Sydney Macarthur, Goldsmith Avenue, PO Box 555, Campbelltown NSW 2560, Australia.

File Submission

9. If your file size is large, ***submit Ziped version of it.***

10. ***Upload the files through the Workshop Web page at:*** http://fistserv.macarthur.uws.edu.au/icse2000-webe/icse2000-submission.htm

Submission Deadline: 26 April 2000

Checklist -- Requirements

- Two soft copies of the full manuscript one in PDF and the other in RTF.
- Soft copy of the Abstract
- Copyright form by mail

Exhibit 3

Untitled

From y.deshpande@uws.edu.au Sat Apr 1 23:29:59 2000
X-VM-v5-Data: ([nil nil nil nil nil nil nil nil nil]
["714" "Sun" "2" "April" "2000" "17:30:21" "+1000" "Yogesh Deshpande" "y.de
shpande@uws.edu.au" nil "26" "Paper submission for the WWW9 Web Engineering Worksho
p" "^From:" nil nil "4" "2000040207:30:21" nil nil nil]
nil)
Return-Path: y.deshpande@uws.edu.au
X-Lotus-FromDomain: IBMUS
Mime-Version: 1.0
Content-Disposition: inline
From: Yogesh Deshpande <y.deshpande@uws.edu.au>
To: savitha@almaden.ibm.com, dulce@almaden.ibm.com, arnon@angi.almaden.ibm.com,
bblanch@almaden.ibm.com, petkovic@almaden.ibm.com
cc: Yogesh Deshpande <y.deshpande@uws.edu.au>, s.murugesan@uws.edu.au
Subject: Paper submission for the WWW9 Web Engineering Workshop
Date: Sun, 2 Apr 2000 17:30:21 +1000

Dear Authors,

Paper title: Engineering the Web for Mutlimedia

As intimated in the previous message, could you take into account the reviewers' comments, reproduced below, to revise your paper and submit a camera-ready copy by 14 April at the latest?

May I emphasise that the deadlines are going to be very tight because the Springer-Verlag publication is timed for a June/July release. Please let me or San know asap if any clarifications are needed.

Here are the reviewers' comments:

Some cosmetic suggestions: the paper is quite dense, which makes it hard when reading. In section 2.1 the points mentioned should have bullet points for easier reading.

See you at the Workshop!

Yogesh Deshpande

Exhibit 4

San MURUGESAN <s.murugesan@uws.edu.au> on 05/09/2000 06:07:29 AM

To: schwabe@inf.puc-rio.br, luiselena@inf.puc-rio.br, fer@sol.info.unlp.edu.ar, gustavo@sol.info.unlp.edu.ar, Savitha Srinivasan/Almaden/IBM@IBMUS, Dulce Ponceleon/Almaden/IBM@IBMUS, Arnon Amir/Almaden/IBM@IBMUS, Brian Blanchard/Almaden/IBM@IBMUS, Dragutin Petkovic/Almaden/IBM@IBMUS, joaoc@dai.ed.ac.uk, dr@dai.ed.ac.uk, Rosa.Carro@ii.uam.es, Estrella.Pulido@ii.uam.es, Pilar.Rodriguez@ii.uam.es, W.Janssen@telin.nl, M.Steen@telin.nl, C.Kerer@infosys.tuwien.ac.at, E.Kirda@infosys.tuwien.ac.at, gaedke@teco.uni-karlsruhe.de, graef@teco.uni-karlsruhe.de
cc: Yogesh Deshpande <y.deshpande@uws.edu.au>, kennethn@acm.org, s.murugesan@uws.edu.au
Subject: WWW9 Web Engineering Workshop Programme and Abstracts of Papers

Dear Authors/Presenters,

1. The Workshop Programme and Abstracts of the Papers/Presentations are attached.

The Workshop begins at 9AM. You may please register early and navigate to the Workshop venue by 8.50AM.

2. The programme, in addition to contributed presentations, includes:

Keynote address
Improving Web Performance and Efficiently Serving Dynamic Content at Highly Accessed Web Sites
Arun Iyengar, IBM T.J. Watson Research Center, USA

Panel Discussion
Roles of Academia, Practitioners and Professional Bodies in Advancing Web Engineering

3. PRESENTATION: Duration for each presentation would be 20 minutes, INCLUDING DISCUSSIONS. Please limit your formal presentation to 15 minutes or less to allow enough time for questions and answers and discussions.

4. CONFIRMATION: Please confirm presentation of your paper and inform the name of the presenter, by sending an EMAIL to s.Murugesan@uws.edu.au AT THE EARLIEST.

5. If you have not yet registered for the Workshop, you may please register as soon as possible. See: http://www9.org

We look forward to seeing you soon.

Regards

San Murugesan, Yogesh Deshpande and Kenneth Norton
Workshop Co-Chairs

Dr San Murugesan
Dept of Computing and Information Systems
University of Western Sydney Macarthur
Campbelltown NSW 2560; Australia
Phone: +61-2- 4620 3513
Fax: +61-2- 4626 6683

email: s.murugesan@uws.edu.au
web page: http://fistserv.macarthur.uws.edu.au/san/

 - WWW9-WebE-program.PDF

 - WWW9-Abstracts-Proc.PDF

Exhibit 5

Third WWW9 Workshop on
Web Engineering
Amsterdam, 15 May 2000

Session S1: 9.00 AM – 10.30 AM

Welcome

Introduction to the Workshop

Introduction by Participants

Keynote
Improving Web Performance and Efficiently Serving Dynamic Content at Highly Accessed Web Sites
Arun Iyengar, IBM T.J. Watson Research Center, USA

10.30 – 11.00 ***Coffee Break***

Session S2: 11.00AM – 12.30 PM

Layout, Content and Logic Separation in Web Engineering
Clemens Kerer and Engin Kirda
Distributed Systems Group, Technical University of Vienna, Austria

Improving Web-Site Maintenance With TANGOW by Making Page Structure and Contents Independent
Rosa María Carro, Estrella Pulido, Pilar Rodríguez
Escuela Técnica Superior de Informática, Universidad Autónoma de Madrid, Madrid, Spain

Engineering the Web for Multimedia
Savitha Srinivasan, Dulce Ponceleon, Arnon Amir, Brian Blanchard, Dragutin Petkovic, IBM Almaden Research Center, San Jose, USA

Rapid Service Development: An Integral Approach to E-business Engineering
Wil Janssenand Maarten Steen, Telematics Institute, P.O. Box 589, 7500 AN Netherlands

---

12. 30 – 2.00 Lunch Break

Session S3 2.00– 3.30 PM

Development and Evolution of Web-Applications using the WebComposition Process Model
Martin Gaedke, Guntram Gräf
Telecooperation Office (TecO), University of Karlsruhe, Germany

Web Design Frameworks: An approach to improve reuse in Web applications
Daniel Schwabe *, Gustavo Rossi **, Luiselena Esmeraldo *, Fernando Lyardet**
*Departamento de Informática, PUC-Rio, Brazil
**LIFIA Facultad de Informática. UNLP, La Plata, Argentina

Synthesis of Web Sites from High Level Descriptions
Joao Cavalcanti and David Robertson
Institute for Representation and Reasoning - Division of Informatics, The University of Edinburgh

---

3.30 – 4.00 PM Coffee Break

Session S4 4.00 – 5.00 PM

Panel Discussion
Roles of Academia, Practitioners and Professional Bodies in Advancing Web Engineering
Chair: Kenneth Norton, Twelve Entrepreneuring, USA

Third Workshop on Web Engineering
World Wide Web Conference, May 2000
Amsterdam

Abstract of Papers/Presentations

Keynote
Improving Web Performance and Efficiently Serving Dynamic Content at Highly Accessed Web Sites
Arun Iyengar
IBM T.J. Watson Research Center
aruni@us.ibm.com
http://www.research.ibm.com/people/i/iyengar/

This talk presents techniques for designing Web sites which need to handle large request volumes, provide high availability, or generate significant dynamic content. These techniques include using load balancers to distribute load among multiple Web servers, Web server accelerators, and caching dynamic pages. We present the architecture of a scalable and highly available Web server accelerator we have developed which significantly improves Web server performance. We also present a publishing system we have developed for efficiently generating complex dynamic pages from simpler fragments. The talk will also describe new techniques we have developed for keeping cached dynamic data current and synchronizing caches with underlying databases.

The work described in this talk has been used to improve performance at several highly accessed Web sites and been incorporated into several products. We will describe how our publishing system is being used at the official Web site for the 2000 Olympic Games. We will also illustrate how a slightly earlier version of our technology was deployed at the official Web site for the 1998 Olympic Winter Games. Performance and high availability were critical for this site not only because it was one of the most popular on the Web but also because the site was providing results which were constantly changing. The 1998 Olympic Games Web site achieved quick response times even under peak loads which set world records and was available 100% of the time.

Layout, Content and Logic Separation in Web Engineering

Clemens Kerer and Engin Kirda
Distributed Systems Group, Technical University of Vienna, Austria
Argentinierstrasse 8/184-1, A-1040 Wien, Austria
{C.Kerer, E.Kirda}@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at/

The rapid development of flexible, layout independent web sites is an increasingly important problem. Flexibility, scalability and the ability to adapt to evolving layout requirements is a key success factor for many web sites. A fundamental way to meet these requirements is to strictly separate business logic from the layout and the content. The World Wide Web Consortium's XML and XSL standards aim at providing the separation between layout and content only. In this paper, we describe our ongoing work in separating the layout, the content and the logic of web sites and show how this separation is supported by the tool MyXML. The underlying concepts of our solution are a declarative description of the layout information, automatic generation of static and dynamic pages and support of interconnection to extended information sources such as databases.

Improving Web-Site Maintenance With TANGOW by Making Page Structure and Contents Independent

Rosa María Carro, Estrella Pulido, Pilar Rodríguez
Escuela Técnica Superior de Informática, Universidad Autónoma de Madrid,
Campus de Cantoblanco, 28049 Madrid, Spain
{Rosa.Carro, Estrella.Pulido, Pilar.Rodriguez}@ii.uam.es

In this paper we discuss how Web site maintenance can be improved by making page structure and contents independent. This is the principle used in TANGOW (Task-based Adaptive learNer Guidance On the Web) for designing and maintaining Web-sites. In TANGOW, the information structure and contents are managed independently, what facilitates the maintenance tasks. It also allows the specification of a common design pattern for all the final HTML pages, which the system dynamically generates starting from static HTML pieces. In the first section we introduce some common issues and problems in Web-site development. The TANGOW system is presented in the second section, focusing on those characteristics that make it useful for the development of Web-sites. Based on a real world example, the next section shows how TANGOW helps to solve the problems mentioned in section 1. Finally, some conclusions are taken from the described application

Engineering the Web for Multimedia

Savitha Srinivasan, Dulce Ponceleon, Arnon Amir, Brian Blanchard, Dragutin Petkovic
IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120 USA
{savitha, dulce, arnon, bblanch, petkovic}@almaden.ibm.com

This paper examines the issues related to developing web applications that use digital media, with particular emphasis on digital video. The nature of digital video brings additional complexity to engineering solutions on the web due to the large data sizes in comparison with text, the temporal nature of video, proprietary data formats, and issues related to separation of functionality between content creation, content indexing with associated metadata, and content delivery. The goal of this paper is to contribute to the understanding of different component technologies involved in deploying video-based web applications, and the tradeoffs involved with each option. As an illustrative example, we describe the requirements leading to the architecture of a video-based web application, CueVideo: a system for search and browse of video and related material.

Rapid Service Development: An Integral Approach to E-business Engineering

Wil Janssen and Maarten Steen
Telematics Institute, P.O. Box 589, 7500 AN Netherlands
{W.Janssen,M.Steen}@telin.nl

Developing e-business solutions is a complicated task. It involves many different disciplines and requires knowledge of e-business technologies as well as business processes. In this paper we look into what causes this complexity, and discuss an approach to overcome the current barriers in e-business engineering. The approach combines knowledge of processes and technology with a new e-business engineering methodology, called Rapid Services Development (RSD). The ingredients of RSD are discussed in detail, and linked to current engineering approaches.

Development and Evolution of Web-Applications using the WebComposition Process Model

Martin Gaedke, Guntram Gräf
Telecooperation Office (TecO), University of Karlsruhe, Vincenz-Priessnitz Str. 1, D-76131 Karlsruhe, Germany
E-Mail: {gaedke|graef}@teco.uni-karlsruhe.de

From a software engineering perspective the World Wide Web is a new application platform. The implementation model that the Web is based on makes it difficult to apply classic process models to the development and even more the evolution of Web-applications. Component-based software development seems to be a promising approach for addressing key requirements of the very dynamic field of Web-application development and evolution. But such an approach requires dedicated support. The WebComposition Process Model addresses this requirement by describing the component-based development of Web-applications. It uses an XML-based markup language to seamlessly integrate with existing Web-standards. For the coordination of components the concept of an open process model with an explicit support for reuse is introduced. By describing application domains using domain-components the process model addresses the need for a controlled evolution of Web applications.

Web Design Frameworks: An approach to improve reuse in Web applications

Daniel Schwabe *, Gustavo Rossi **, Luiselena Esmeraldo *, Fernando Lyardet**
*Departamento de Informática, PUC-Rio, Brazil
{schwabe, luiselena} @inf.puc-rio.br
**LIFIA Facultad de Informática. UNLP.
La Plata, Argentina
{fer,gustavo}@sol.info.unlp.edu.ar

In this paper we introduce Web design frameworks as a conceptual approach to maximize reuse in Web applications. We first discuss the need for building abstract and reusable navigational design structures, exemplifying with different kinds of Web Information Systems. Then, we briefly review the state of the art of object-oriented application frameworks and present the rationale for a slightly different approach focusing on *design* reuse instead of *code* reuse. Next, we present OOHDM-frame, a syntax for defining the hot-spots of generic Web application designs. We illustrate the use of OOHDM-frame with a case study in the field of electronic commerce. We finally discuss how to implement Web design frameworks in different kind of Web platforms.

Synthesis of Web Sites from High Level Descriptions

Joao Cavalcanti and David Robertson
Institute for Representation and Reasoning - Division of Informatics -
The University of Edinburgh
joaoc@dai.ed.ac.uk dr@dai.ed.ac.uk

As use of Web sites has exploded, large amount of effort have gone into the deployment of sites but little thought has been given to methods for their design and maintenance. This paper reports some encouraging results on the use of automated synthesis, using domain-specific formal representations, to make design more methodical and maintenance less time consuming.